INFO 5100 Project 3 Report

Jiahan Yan (jy468)
Jiamin Zeng (jz863)
Yijiang Wang (yw774)
Kyle Griffin (kmg272)

Purpose and Story

The main purpose behind our project is to provide a visualization for an interesting and peculiar property of Wikipedia. If you navigate to a Wikipedia article, click on the first hyperlink in the article (the first non-parenthesized hyperlink in the introduction), and then repeat this process for each subsequent page, eventually you will end up on the Wikipedia page for “Philosophy” roughly 95% of the time.

This characteristic is a result of the classification-based nature of Wikipedia articles. The first sentence of each page is an introductory definition of the subject; inherently, when defining a topic we typically use nouns that are slightly less specific than the topic we are describing. By continuously applying this rationale and ‘going up one level of abstraction’ for each topic we encounter, eventually we end up describing Philosophy - the fundamental understanding of thought and reality. As an example, say we wanted to explain “d3.ScaleLinear()”. One way to represent this description chain is:

d3.ScaleLinear() is a d3.js function call →
d3.js is a JavaScript library →
JavaScript is a programming language →
a programming language is a formal language →
a formal language is a set of symbols →
symbols are a concept in logic →
logic is the study of reason →
the study of reason is a fundamental component of philosophy

The actual “wiki trick” generally follows a more convoluted route but almost always ends up in the same place.

For our project, we wanted to display this connection chain between pages not only for one topic, but for several different topics. By including many different topics, we were able to visualize not only the topics’ connections to Philosophy, but also the classification hierarchy between these topics (i.e. the places where different topics converge before ultimately leading to Philosophy - “JavaScript”, “python” and “C++” would all converge at “Programming Language” before continuing on to Philosophy along the same path).

Data Processing

We had two types of data that we used in our visualization: the different topics we wanted to include as the end nodes in our visualization (i.e. article titles), and the text data from the Wikipedia pages themselves.

We manually generated our topic listings and stored them in text files, but in order to generate the Wikipedia HTML data, we scraped the wiki articles for each topic. Our crawler is written in python and incorporates the Beautiful Soup parsing tool. We treat each wiki article as a “node” and the hyperlink from one article to another as a “path”. Using these nodes and paths, our script constructs a tree in which the root is philosophy and the leaves are the initial topics in our listing.

In detail, the logic flow is: request a starting page → parse the html → keep effective urls → go to next based on the first URL encountered in the article article → repeat until we hit philosophy. In this process we record the nodes and paths, dump them into to a JSON file at last and feed that JSON to the d3 library to generate our visualization.

As we were employing scraping techniques to pull somebody else’s data, we had to consider the ‘ethics’ of our methodology as discussed in class. We designed our implementation such that the Wikipedia data for each topic would only have to be scraped once and would then be stored in a JSON file for later use. This resulted in a fairly low number of overall requests. Additionally, we reasoned that our scraping tools would not be run frequently (we were creating a private project, rather than a public webpage), and that an large organization like Wikipedia has the infrastructure in place to comfortably handle infrequent bursts of a few hundred requests.

Design and Implementation

In constructing our visualization, we adapted elements from Mike Bostock’s Radial Tidy Tree implementation of d3.tree(). A radial tree is a representation of a tree structure with the head at the center of a circle, and the nodes/leaves all branching out from that central root. In our situation we were dealing with many different branches that have an unknown number of levels. This made a traditional tree unsuitable and impractical, as it would likely either force the user to scroll to see data, or for us to implement a specialized zooming functionality (very possible, but not ideal if there is a better alternate workaround). We decided to take a different approach and felt that the radial tree was a efficient and clean way of visualizing our data, while bypassing the constraints of a traditional tree.

Each small circular element in the tree represents a different Wikipedia page, and if a path exists between two nodes that indicates that the innermore node was the first hyperlink in the article body of the outermore node. The concentric rings signify incrementing levels of separation away from the Philosophy page. These rings serve as a reference tool for the user, as otherwise it can be difficult to judge the the lengths of longer paths.The titles of articles represented by nodes can be seen by mouse hovering over the node, and a user can also click

on a each node in the tree and quickly see the full list of pages that lie on that topic's path to Philosophy on the right. Lastly, a user is able to click on each of our four different topic categories, which filters out the topics from that category (all categories are enabled by default). We used different colors to represent each of these categories, which is reflected in the filter color, the end node color, and the path color when an end node is selected.

Work done by each member:

Jiahan Yan (jy468): wrote crawler script, assist visualization implementation
Kyle Griffin (kmg272): researched and assisted with data management and design, drafted final report
Yijing Wang (yw774): built Tree Visualization part and merge the code written by other members.
Jiamin Zeng (jz863): researched hover effect and did some design part related to CSS.